LEARNING SPACE TIME DYNAMICS WITH PDE GUIDED NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Spatio-Temporal processes bear a central importance in many applied scientific fields. Generally, differential equations are used to describe these processes. In this work, we address the problem of learning spatio-temporal dynamics with neural networks when only partial information on the system's state is available. Taking inspiration from the dynamical system approach, we outline a general framework in which complex dynamics generated by families of differential equations can be learned in a principled way. Two models are derived from this framework. We demonstrate how they can be applied in practice by considering the problem of forecasting fluid flows. We show how the underlying equations fit into our formalism and evaluate our method by comparing with standard baselines.

1 INTRODUCTION

We consider the problem of learning complex dynamics typical of physical systems using deep learning models. Modeling these dynamics is central to almost any problem in applied physics. The classical approach to this problem is model based : the dynamics of the physical system is usually formulated using Partial Differential Equations (PDEs) describing the evolution of state variables. PDEs are then solved through some clever numerical scheme (as in Meerschaert & Tadjeran (2006) for example). This methodology is used in many application domains like climate science, financial engineering, etc.. Although it has been extensively developed, several challenges remain open. Modeling physical phenomena accurately can be tedious, if not unfeasible : the relations between state variables are often complex and, even when those are known, specifying a functional form for the dynamics can be difficult (Lions et al. (1992), Peebles (1980)). Moreover, designing accurate, robust and computationally feasible numerical schemes for PDEs is usually a difficult problem for which current solutions rely on extensive domain specific background.

The availability of huge data quantities gathered from different types of sensors or simulations is a strong incentive to develop data driven approaches, as an alternative or complement to standard model based methods. For many real world problems, the underlying physics of the phenomenon is not always known in its entirety and data-driven methods, inspired by our current physical knowledge, appear as a particularly promising alternative to model based approaches, see Bezenac et al. (2018), Ling et al. (2016), Stewart & Ermon (2016) or Rudy et al. (2017) for some examples. On the other hand, mature fields such as differential equations could provide machine learning with useful insights and intuitions. We advocate here a ML approach incorporating structural priors inspired from such a well developed domain.

We consider the problem of learning dynamical systems where available observations only provide partial information on the system state and dynamics which is a typical scenario in real world applications, be it in weather forecasting, financial engineering, or even video prediction. The objective is to learn from such observations the underlying system dynamics so as to 1. forecast future observations, 2. learn states and dynamics that follow as much as possible the laws of physics. We focus on data generated by complex highly non-linear differential equations and analyze whether these dynamics can be learned by Deep Learning models. This context is particularly interesting since simulations relying on PDEs are essential in many applied disciplines and since this allows us to perform controlled experiments providing access to all the variables of the problem.

Our contributions are the following :

- We propose a general framework for modeling dynamical systems governed by PDEs with neural networks and we derive two instances. This framework is inspired from the standard physical approach which has proven to be powerful to model complex real-world phenomena. Many forecasting problems can be cast into this framework, although this goes beyond the scope of this paper.
- As a case study, we consider two PDEs describing the evolution of fluid flow, namely the *Euler* and *Navier-Stokes equations*, and show how they can be formulated within this formalism. We study the performance and behaviour of our models on the problem of forecasting fluid flow.

We introduce our framework in section 2, detail the framework and its instances in section 3 and show how PDEs fit within this framework in section 4. Experimental results and analysis are described in section 5 and recent related work is reviewed in section 6.

2 BACKGROUND AND PROBLEM SETTING

2.1 LEARNING FROM PARTIALLY OBSERVED DATA

A dynamical system in our context can be broadly defined as a function X with values in Ω which obeys a set of differential equations. We consider spatio-temporal dynamics for which X can be written as a function of $(t, x) \in \mathbb{R} \times \mathbb{R}^d$ where t and x are respectively the time and space variables. The spatial vector-valued function X_t contains the quantities of interest describing a studied physical system at time t and gives the evolution in time and space of the system's state. More precisely, we will be interested in differential equations which can be written as :

$$\frac{dX_t}{dt} = F(X_t, \mathcal{D}_x X_t) \tag{1}$$

where X_t denotes the state vector at time t, $\mathcal{D}_x X_t$ the spatial derivatives of X_t and F is a function with values in Ω (common instances for F involve the gradient operator ∇X, the laplacian $\nabla^2 X$,...). By specifying a particular structure on F, one can describe the dynamics of many different classes of systems : an important example is that of the Navier-Stokes equations from fluid dynamics which bear a central importance in many applied sciences and which will be our application example.

2.1.1 NEURAL IMPLEMENTATION OF NUMERICAL SCHEMES

Solving differential equations usually requires using a numerical scheme in order to approximate the solution. Finite difference methods for example make use of discrete approximations of the derivatives. Let us suppose available a sequence of states $\{(X_t)_t\}$, each X_t being a function of the space variable x, X_t and $X_{t+\delta t}$ being separated by a short time interval δt for all t. A straightforward discretization of equation 1 is the forward Euler scheme :

$$X_{t+\delta t} - X_t \approx \delta t \cdot F(X_t, \mathcal{D}_x X_t)$$

wher δt is the discretization timestep.

Using a parameterized neural network r_θ, we can train it in order to fit the relation above :

$$X_{t+\delta t} \approx X_t + \delta t \cdot r_\theta(X_t, \mathcal{D}_x X_t)$$

This is reminiscent of the skip connection structures used in the residual block of ResNet or in some RNN models. Linear multi-step methods extend this idea by performing several integration steps. A simple multi-step extension of the Euler scheme could then be :

$$X_{t+k\delta t} \approx X_t + \delta t \sum_{l=0}^{k-1} r_\theta(X_{t+l\delta t}, \mathcal{D}_x X_{t+l\delta t})$$

This sequential process could be implemented by a ResNet architecture so that the latter could be used as a parametric model for approximating solutions of differential equations.

Moreover, spatial derivatives of X_t at a given point x can classically be expressed with finite difference schemes over the values of the function at neighbouring points V_x :

$$\mathcal{D} X_t(x) = K(\{X_t(x') \mid x' \in V_x\})$$

where K is a linear function. Thus, we can express each component of $\mathcal{D}_x X_t$ as a convolution over X_t which motivates the use of Convolutional networks. Thus, given a sufficiently large training set of state sequences and supposing that we are able to learn its dynamics, we could, just by knowing the initial state of our system, use the learned operator and forecast its behaviour for any time horizon.

2.1.2 LINKING STATES TO PARTIAL OBSERVATIONS

In realistic settings, the state is generally only partially observed *e.g.*, when studying the ocean's circulation, variables contained in the system's state such as temperature or salinity are observable while others such as velocity or pressure are not. In other words, the measured data is only a projection of the model complete state X. We model this measurement process with a fixed operator $\mathcal{H}$ linking the system's state X_t to the corresponding observation Y_t :

$$Y_t = \mathcal{H}(X_t)$$

To use the former example, if only the surface temperatures are observed, the operator $\mathcal{H}$ will select among all the possible oceanic variables the ones related to sea surface temperature. In the following, $\mathcal{H}$ is supposed to be known and fixed.

3 FRAMEWORK

To summarize, observations Y_t are supposed to be generated by a state space model with variable X_t and the main objective is the prediction of future observations, e.g. Y_{t+k} for horizon k. The general form of the model for a prediction at horizon 1 is then :

$$\begin{cases} \text{initial state } X_0 \\ X_{t+1} = g(X_t) \\ \quad Y_t = \mathcal{H}(X_t) \end{cases} \tag{2}$$

In this expression, prior information comes from the projection function $\mathcal{H}$ and from the state dynamics embedded in g. The classical approach in environmental science is *data assimilation* : starting from an initial estimate of the state at time t, a PDE numerical scheme is used to infer state at time $t+1$, $\hat{X}_{t+1}$ and then $\hat{Y}_{t+1}$. The initial state estimate is a function of past information. This approach relies on an explicit physical model expressed as a PDE for solving an inverse problem : infer X_t from observations Y_{t-k}^t. The model in 2 could be alternatively implemented via a recurrent autoregressive NN architecture where past inputs are used together with the current state in order to predict the next state. We will use a variant of this formulation where the auto-regressive component and the dynamic model are separated, which is not the case with a generic RNN structure.

We make the simplifying hypothesis that the information loss through the projection $\mathcal{H}$ is so that a state X_t can be deterministically reconstructed using a long enough sequence. In other words, there exists an integer k and an operator $\mathcal{E}$ such that :

$$X_t = \mathcal{E}(Y_{t-k+1}, ..., Y_t)$$

Let us consider, to simplify, one step predictions : given a sequence of initial observations $(Y_1, ..., Y_k)$, our main objective is to find the best estimation $\widehat{Y}_{k+1}$ for the observation Y_{k+1} where observations Y are supposed to be generated an underlying process X. The proposed model proceeds in three steps :

- Prediction of the current state : Given past observations, we use a learnable operator e_ω parameterized by ω to produce a (complete) state vector $\widetilde{X}_t$.
- Prediction of the future state : $\widetilde{X}_t$ is given as input to a second learnable operator f_θ parameterized by θ which produces the next state $\widehat{X}_{t+1}$.
- Projection of the future observation : $\widehat{X}_{t+1}$ is mapped onto the space of observations using operator $\mathcal{H}$ to predict future observation $\widehat{Y}_{t+1}$.
  This is summarized in the following system :

$$\begin{cases} \widetilde{X}_t = e_\omega(Y_{t-k+1}, ..., Y_t) \\ \widehat{X}_{t+1} = f_\theta(\widetilde{X}_t) \\ \widehat{Y}_{t+1} = \mathcal{H}(\widetilde{X}_{t+1}) \end{cases} \tag{3}$$

The learning problems amounts to optimizing the following loss :

$$\min_{\theta,\omega} \mathbb{E}_{(Y_1,...,Y_{k+1})\in \text{Data}}[d(\mathcal{H}(f_\theta(e_\omega(Y_1, ..., Y_k))), Y_{k+1})] \tag{4}$$

where d is the loss metric and e and f are functions belonging to appropriate parametric families. Time lag parameter k can be set up by cross validation. As discussed in section 2, a natural choice for f is a convolutional residual network modeling the system's dynamics whereas for e designing the right family of functions will depend on the structure of the studied data.

3.1 Forecasting models

Let us now introduce two instances of the above model. For simplification we consider first one step prediction and then introduce extensions to multistep predictions in section 3.2.

3.1.1 Joint training of estimation and forecasting

When the information about the underlying process is limited to observations, the straightforward way to solve the minimization problem equation 4 is to train e and f jointly. Prior information here is limited to the architectures of e and f as well as that of the pipeline equation 3 and the fixed operator $\mathcal{H}$. Let $\left\{\left((Y^{(i)}_{t-k+1}, ..., Y^{(i)}_t), Y^{(i)}_{t+1}\right)\right\}_i$ be a training set of observation sequences, each sequence being identified by index i. The loss corresponding to sequence i is then :

$$\mathcal{L} = d\big[\mathcal{H}(f_\theta(e_\omega(Y_{t-k+1}, ..., Y_t))), Y_{t+1}\big]$$

With this approach only observations are needed for training without any additional supervision. On the other hand, there is no reason for it to have any interpretable structure or correspond to the physical modelling of the studied system.

3.1.2 Pretraining the state estimator

Often some form of additional knowledge on the system state is available and can be exploited.

- It is sometimes possible to access to a limited number of examples of state sequences through costly measurements. Then pretraining of e and f could be beneficial, its success shall depend on the complexity of the studied dynamics and on the loss of information through $\mathcal{H}$.
- More often it is possible to have access to a noisy or approximate version of the states X_t through simulations. Complex simulators and generated data are available in several domains : it is common for models to be developed using simulated data and then to be adapted to real observations. A lighter alternative is to use a simplified model which allows fast and cheap simulations. Generated sequences can then be used e.g. for pretraining e and f before fine-tuning with real observations or more complex simulations. In both cases, the simplified dynamics plays the role of a prior over the underlying process. In the experiments in section 5.2, we will use a simplified dynamical model (Euler) as a prior for a more complex one (Navier Stokes) in order to illustrate and test this idea.

3.2 Multi-step forecasting

We present below two strategies that could be used for prediction at horizon $l > 1$. In order to forecast $l \geqslant 2$ steps ahead, once we have the estimated state at time t, $\widetilde{X}_t$, given by e_ω, we may apply one of the following strategies :

- Single State Estimation (**SSE**) : the estimation operator e is used only once to give the forecasting operator an initial state $\widetilde{X}_t$:

$$\widehat{X}_{t+l} = (f_\theta \circ \cdots \circ f_\theta)(\widetilde{X}_t)$$

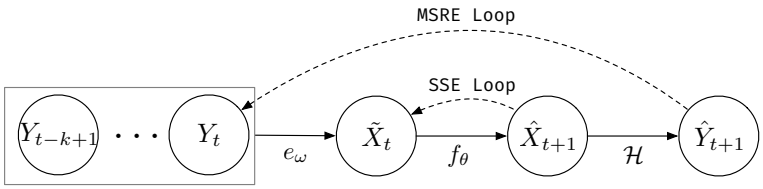

Figure 1: Illustration of the two multi-steps prediction strategies - SSE and MSRE - introduced in section 3.2.

- Multiple State REstimation (**MSRE**) : We use the forecast operator to produce the next state
$$\widehat{X}_{t+1} = f_\theta(\widetilde{X}_t)$$
Then, for the forecast at $t+2$, we estimate the state at time $t+1$ by plugging the generated observation $\mathcal{H}(\widehat{X}_{t+1})$ in the estimation operator e_ω :
$$\widetilde{X}_{t+1} = e_\omega(Y_{t-k+2}, ..., Y_t, \mathcal{H}(\widehat{X}_{t+1}))$$
so that :
$$\widehat{X}_{t+2} = f_\theta(\widetilde{X}_{t+1})$$
and then iterate for the following steps.

In both cases, observations up to Y_{t+l} will be used as targets in the loss term when learning to predict at horizon l.

Note that using a multi-step forecasting objective during training could be useful since it imposes additional constraints on the learned functions. This will be explored in the experimental section. When using multi-step forecasting at training time, the error is back-propagated through the l prediction steps and one has to choose a principled way of supervising the training.

Moreover, we have also used and compared different sampling strategies for long sequences training : systematic teacher forcing, no teacher forcing and scheduled sampling as presented in Bengio et al. (2015). Quantitative results are presented in the appendix.

When training with long sequences, another problem arises : depending on the distribution of the eigenvalues of the matrices, we can observe weight values either diverging to infinity or going to 0. Following the analysis done in Saxe et al. (2013), we have observed that orthogonal initialization of the weights of f helps solving this issue.

4 Examples from fluid dynamics : Euler and Navier-Stokes

As an illustration, we now focus on the problem of the evolution of a two-dimensional fluid mechanics system. The equations governing fluids are among the most studied. One of the challenging aspects of those highly non-linear equations is the chaotic behaviour of turbulent flows. We use, as examples, instances of Euler and Navier-Stokes equations which are briefly introduced below. More information is available in the appendix and Acheson (1989) is a comprehensive reference textbook. Euler being a simpler version of Navier Stokes, only the latter is presented below.

Navier Stokes equations for incompressible fluids are :

$$\begin{cases} \dfrac{\partial u}{\partial t} + (u \cdot \nabla)u = -\dfrac{\nabla p}{\rho} + g + \nu \nabla^2 u \\ \dfrac{\partial \rho}{\partial t} + (u \cdot \nabla)\rho = 0 \\ \nabla \cdot u = 0 \end{cases} \tag{5}$$

The first equation results from Newton's second law, the second comes from the conservation of density while the third is the incompressibility condition. In those equations, u denotes a two-dimensional velocity field, p is a pressure scalar field, ρ is the fluid density and g is the gravitational force. $\nabla \cdot u$ is the divergence of velocity u, $u \cdot \nabla$ is the advection operator $u \cdot \nabla = u_x \dfrac{\partial .}{\partial x} + u_y \dfrac{\partial .}{\partial y}$

Euler equations are similar to Navier Stokes but do not consider the particular nature of the fluid and ignore the viscosity term $\nu\nabla^2 u$ in the first equation of (5) by taking $\nu = 0$.

For Navier-Stokes, ν is the *kinematic viscosity* and ∇^2 is the Lagrangian operator. The importance of viscosity in a fluid, and thus a way to assess its qualitative behaviour, cannot be measured through ν alone. The most commonly used quantity to describe it is the *Reynolds number* R. This is the ratio of the inertia and the viscosity terms. If R is small, it means that the fluid is very viscous, behaving more like honey ; If R is large, the viscosity term is small and the fluid behaves more like water. A definition of the *Reynolds number* is provided in the appendix.

Equation (5) is not of the same form as equation 1 as we still have the pressure variable p as well as the null divergence constraint. It is however possible to show that it follows the general form of equation 1 as there exists an operator $\mathbb{P}$ which outputs divergence-free vector fields so that, for Navier Stokes, the third equation is dropped and the first equation takes the following form :

$$\frac{\partial u}{\partial t} = \mathbb{P}[\int -(u \cdot \nabla)u + \nu\nabla^2 u]$$

This is similar to equation 1 so that we can put the whole system in that form with $X = (\rho, u)$. This means that all the results and developments in previous sections apply for this family of fluid equations. We give more details about this in the appendix[1]

Note that in what follows, we have $\mathcal{H}(X) = \rho$. In other words, we observe the scalar field of densities and the complete state is obtained by concatenating the 2D velocity flow u to ρ.

5 EXPERIMENTS

The architecture of e_ω is based on the Unet presented in Ronneberger et al. (2015) which is a convolutional-deconvolutional network with skip connections which takes as input k feature maps corresponding to the k different observations. In all the following experiments, we have chosen $k = 3$. For the forecasting operator f_θ, we use the ResNet architecture He et al. (2016) with Leaky ReLU non-linearities of parameter 0.1 and 6 residual blocks.

Our data is generated from a numerical simulation, using the Mantaflow fluid simulation library Thuerey & Pfaff (2018). We study three different dynamics : the first one is governed by the Euler equations, the second and the third by the Navier-Stokes equations with $R = 5000$ and $R = 100000$. The former corresponds to a viscous fluid with complex dynamics while the latter corresponds to a low viscosity and is supposed to have a simpler dynamics. For each setting, we produce 300 training sequences of 500 timesteps each (one timestep being of duration $0.5s$), randomly selecting initial conditions for each sequence. We then subsample 5 times in time (two successive states are then separated by $2.5s$). In order to tune the hyperparameters of our experiments, we also generate a validation set of 200 samples as well as a test set of 100 samples.

The choice of a normalization strategy for the data has not been immediate. We finally settled for the normalization by dividing each component of the state by its empirical standard deviation over the dataset. We do not set the mean to zero as the positivity of the density is an important constraint of the problem. As for the baselines, we have been using 1) a simple ResNet which takes 3 past observations as input to produce a future observation directly (for multi-step forecasting, we just give the produced observations back to the ResNet in an auto-regressive manner), and 2) a Convolutional LSTM with 2 layers (we have tried LSTMs with different numbers of layers and hidden dimensions and kept the one with the best validation score).

5.1 EULER EQUATIONS DYNAMICS

In this section, we study the test results for the different algorithms for dynamics driven by the Euler equations. Table 1 compares the models to the two baselines ResNet and ConvLSTM. All make use of a training horizon $l = 8$. JT SSE clearly diverges. This is an overfitting problem that we have not been able to solve. SSE acts as an encoder-decoder architecture: all relevant information about

[1] Let us note that, while this might just seem like a mere technicality, it still provides us with an important information : forecasting the dynamics of those equations while following equation 1 automatically makes us find a null-divergence velocity vector field.

	t_0+1	t_0+5	t_0+10	t_0+15	t_0+20	t_0+25	t_0+30
ResNet	0.042	0.106	0.195	0.267	0.324	0.370	0.407
ConvLSTM	0.16	0.24	0.35	0.47	0.57	0.70	0.90
PT MSRE	0.010	0.071	0.102	0.257	0.324	0.377	0.419
PT SSE	0.011	0.059	0.102	0.179	**0.234**	0.288	0.341
JT MSRE	**0.007**	**0.044**	**0.10**	**0.176**	0.235	**0.284**	**0.327**
JT SSE	0.308	0.599	0.594	1.27	5.24	15.4	33.5

Table 1: Test average MSE for two baslines (resNet and ConvLSTM), and 4 instances of the proposed model for different forecasting horizons (columns). Training horizon is $l = 8$. PT and JY are respectively the pretrained and jointly trained models, SSE and MSRE the two multi-step training strategies introduced in section 3.2

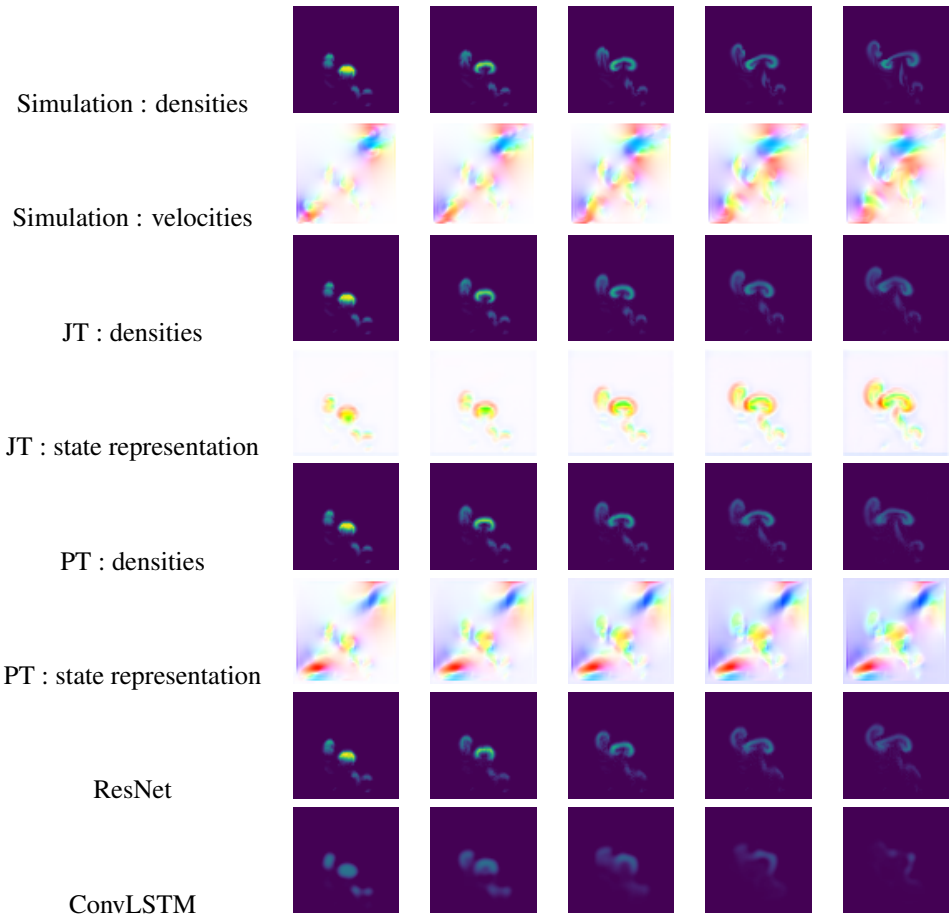

Figure 2: Predicted dynamics for the Euler equations for horizons $t + l = 4, 8, 12, 16, 20$, 1 horizon per column. All algorithms have received as input the same 3 initial images (not shown here). Top 2 rows, correspond to the simulation which in our context is the groundtruth. First row is the generated density and second one is the velocity vector field. For the latter, each point is two dimensional, the color code is provided in the appendix. Rows 3 and 4 give the predictions obtained with a JT MSRE model, rows 5 and 6 the ones for the PT SST one. Bottom rows correspond to the Resnet and the ConvLSTM. Note that there is no velocity state for the ResNet, and the states of the ConvLSTM are not interpretable.

the past observations are encoded into a single initial state, the forecasting operator then generates a whole sequence of future state estimations. This seems to be a hard task without the additional prior given by the structure of the real physical state (which we have in the PT pretrained setting). All

	t_0+1	t_0+5	t_0+10	t_0+15	t_0+20	t_0+25	t_0+30
$l=2$	**0.004**	0.0478	0.195	0.434	0.763	1.20	1.79
$l=5$	0.006	**0.038**	**0.105**	**0.175**	0.239	0.2956	0.345
$l=8$	0.007	0.044	0.10	0.176	**0.235**	**0.284**	**0.327**
$l=11$	0.01	0.052	0.103	0.189	0.245	0.292	0.332

Table 2: Forecasting with Jointly Trained model JT MSRE on Euler simulations: Test average MSE per time-step for different training sequence lengths l, for different forecasting horizons. Training horizon $l = 8$.

	t_0+1	t_0+5	t_0+10	t_0+15	t_0+20	t_0+25	t_0+30
$l=2$	0.03	0.165	0.340	0.5	0.645	0.78	0.912
$l=5$	**0.01**	**0.054**	0.104	0.203	0.287	0.369	0.446
$l=8$	0.011	0.059	**0.102**	0.179	0.234	0.288	0.341
$l=11$	0.012	0.064	0.107	**0.178**	**0.223**	**0.267**	**0.313**

Table 3: Forecasting with Pretrained Estimator on Euler simulations: Test average MSE per time-step for different training sequence lengths l, for different forecasting horizons. Training horizon $l = 8$.

other three algorithms manage to produce good forecasts of the dynamics, even at long prediction horizons ($t_0 + 30$ corresponds to a duration of $75s$). For long horizons, it is to be expected that the PT SSE algorithm should loose its efficiency as all information is supposed to be stored and rolled forward in a markovian fashion while the MSRE algorithms, as well as the ResNet and ConvLSTM baselines, work in an auto-regressive fashion over the observations which makes it easier to use information from the past. However, we can see that PT SSE is still on par with JT MSRE for all time horizons, which shows the efficiency of the true physical representation that the PT alorithm is conditioned on. Those results justify the fact that, from now on, we will only consider the PT SSE and the JT MSRE algorithms.

Figure 2 shows prediction examples from the different models. First of all, let us notice how different the state representations are between the PT and JT instances : while the first presents a structure which is very close to the true physical state, the second has a representation which cannot be interpreted as a velocity field but is rather close to the predicted observation itself. Despite this difference, both JT and ST manage to capture quite well the main features of the observed dynamics. ConvLSTM completely fails and ResNet does quite well even though it misses some subtle changes in density. This highlights the need for skip connections, which are present in the Unet as well as in the ResNet architectures, in order to capture high frequencies. However, let us stress the fact that the studied equations are chaotic in nature which means that even seemingly small errors may induce drastic changes in the evolution of density in the following time-steps and this is indeed what happens in the case of the process shown here[2] : we can actually see it already happening with the evolution of the small features of the density.

Tables 2 and 3 show results for different training horizons l and forecasting horizons, for the two algorithms PT SSE and JT MSRE. At least up to a certain point, the longer the training horizon, the better the long term performance. This is especially true for the PT algorithm as it has to forecast the next states with only one state as input. The improvements observed for the PT algorithm in table 3 are higher than the improvements for the JT algorithm in table 2 and the PT algorithm supersedes JT at longer horizons. This seems to indicate that the state representation with a physical meaning is more stable for long term forecasting. However, we still keep $l = 8$ for the other experiments for computational reasons.

Concerning the sampling method, Teacher Forcing provides the lower performance : it is equivalent to a training horizon of 1. What is more surprising is that No Teacher Forcing and Scheduled Sampling, with a tuned parameter α, have roughly the same performance. A quantitative comparison of the three methods is provided in tables 6 and 7 of the appendix.

[2]One might actually wonder if it makes sense to compare results for very long term horizons as even the simulation will then be diverging from the true dynamic.

	t_0+1	t_0+5	t_0+10	t_0+15	t_0+20	t_0+25	t_0+30
Jointly trained	0.00097	0.0066	0.0196	0.0376	0.0588	0.0824	0.107
Pretrained estimator	0.017	0.108	0.273	0.373	0.446	0.512	0.579
Finetuned pretrained estimator	0.0016	0.0105	0.0256	0.0455	0.0742	0.105	0.169
Jointly trained (half dataset)	0.0015	0.0111	0.0311	0.056	0.083	0.10	0.14
Finetuned pretrained estimator (half dataset)	0.0020	0.010	0.020	0.042	0.096	0.18	0.28

Table 4: Navier Stokes equations with $R = 5000$: Test average MSE per time-step for a Jointly Trained system, a system with pretrained estimator and one with a pretrained estimator finetuned on the Navier-Stokes data, for different forecasting horizons.

	t_0+1	t_0+5	t_0+10	t_0+15	t_0+20	t_0+25	t_0+30
Jointly trained	0.0091	0.047	0.104	0.18	0.24	0.288	0.333
Finetuned pretrained estimator	0.011	0.049	0.103	0.16	0.23	0.297	0.366
Jointly trained (half dataset)	0.0091	0.047	0.10	0.18	0.24	0.29	0.33
Finetuned pretrained estimator (half dataset)	0.011	0.049	0.10	0.16	0.23	0.30	0.37

Table 5: Navier Stokes equations with $R = 100000$: Test average MSE per time-step for a Jointly Trained system and one with a pretrained estimator finetuned on the Navier-Stokes data, for different forecasting horizons, with both systems trained on half the Navier-Stokes training dataset.

5.2 NAVIER-STOKES EQUATIONS DYNAMICS

Let us now consider the more complex dynamics of the Navier-Stokes equations. As before, we use two algorithms: a jointly trained JT MSRE and a pretrained estimator PT SSE, pretrained using Euler equations: Euler dynamics is thus used as a simple structural prior information for the more complex Navier Stokes dynamics. Note that this is clearly an imperfect prior, especially when the viscosity is high (*i.e.* when R is low). We then have a choice between keeping this imperfect estimator, hoping that the forecasting operator will make up for its shortcomings, and fine tuning so that it adapts to the new dynamics. We try the two options, the fine tuning being done by jointly training the whole system, starting with a pretrained estimator, while dividing the learning rate of the estimator optimizer algorithm by a factor of 100.

As table 4 shows, the second option yields clearly better results. We can also see from this table and from table 5 that the JT algorithm still gives good results. In order to further explore the difference in generalization power between the two algorithms, we trained both of them with only half of the training dataset. We can see from table 5 that the loss in performance affects the JT algorithm more than the PT one which becomes comparable to or better than JT. This seems to show that the prior introduced by the pretraining of the estimator does play a role in facilitating the learning, as opposed to the JT where we try to find a good representation from scratch : The PT approach is clearly valuable when training data is scarce. It is also interesting to note that the PT is more competitive when compared to the JT for $R = 100000$, which is to be expected since this is a situation of low viscosity, more comparable to the setting the estimator has been pretrained in. Analyzing the visual outputs of the models (figures are provided in the appendix) for the two viscosity values show that the prediction task is more challenging than with Euler, especially at high viscosity : the models tend to diverge sooner from the ground-truth than for the simpler Euler. Note that, for the PT model, the pre-training has been done with a simpler dynamic which obviously makes the task harder than in the previous section.

6 RELATED WORK

The dynamical system view on NN is not new. In the 90s, several papers developed the analogy between recurrent neural networks and dynamical systems. To quote only a few references Pearlmutter (1995) offers a nice review of state of the art on RNNs and their training algorithms including convergence and stability ; Tsoi & Back (1994) review RNN architectures with local recurrent connections together with their interpretation as non linear IIR/FIR filters ; Haykin (2009) introduces RNNs with regard to state estimation in dynamical systems. More recently, the success of ResNet-like architectures has motivated new developments and interpretations of NNs from a dynamical systems perspective. The interpretation of residual blocks as implementing a simple forward Euler numerical scheme for ODE is highlighted by several authors such as Chang et al. (2018b); Lu et al.

(2018); E (2017). For example, Lu et al. (2018) shows how different ResNet inspired modules can be considered as specific numerical schemes (forward or backward Euler, Runge Kutta, etc..). Chang et al. (2018b) links the depth of ResNet architectures with the size of the discretization step used in ODE. In Gomez et al. (2017); Chang et al. (2018a), the connection with the numerical integration schemes is pushed further by exploiting the idea of reversibility in order to build architectures for which all activations can be reconstructed from the next layer, thus alleviating the need for storing the states of a networks and allowing to build larger architecture . Ruthotto & Haber (2018) develop PDE inspired ResNet like models and establish stability results. This line of work concerns networks for static data and targets the development of alternative architectures or training algorithms.

For dynamic data, Kim et al. (2018) proposes NN models for fluid simulations for computer graphics applications. The objective is to accelerate fluid simulation w.r.t. classical solvers. They learn to generate 2D and 3D fluid velocities (corresponding to states in our examples), using a fully supervised approach. This shares some technical similarities with our own work, but the approach and objectives are clearly different. A fully supervised NN implementing a multistep numerical scheme is also used in Long et al. (2018) for learning PDE from simulations. Raissi (2018) develops a NN framework for learning PDEs from data. In this work either the form of the PDE or the variable dependency is supposed to be known and the approach is again fully supervised.

7 Conclusion

In this work, we address the problem of forecasting spatio-temporal dynamics governed by differential equations where information on the system state comes from incomplete observations.

We have presented a general framework separating the task in two steps, state estimation and forecasting, and proposed a few instances of this framework in order to accommodate for the indetermination in the estimation step. This approach is tested on families of equations governing the dynamics of fluids which are known to be challenging. We have explored empirically some of the properties of the algorithms and evaluated its performance and behavior on high resolution simulated data. We have been able to show that imposing physical priors over the estimator, by pre-training it on simple dynamics then fine-tuning on the target equations, can be beneficial, especially when data is scarce, while jointly training the system without any supervision gives surprisingly good results, especially when lots of data are available.

There are many ideas to explore in order to extend the results. Exploring which additional prior physical constraints could be beneficial to the training and to the discovery of physically meaningful models is an important issue. We also still face difficulties when trying to capture long-term velocity dynamics in the pre-trained setting and this has to be improved if we want to achieve a better performance. Our approach has to be tested on other families of differential equations. Furthermore, while we have been only considering simulated data until now, we must obviously aim for real data.

As a final note, while we have only considered our physically inspired models for physics motivated datasets, we think that the insights provided by the different settings empirically studied in this work might prove useful for other tasks regarding the forecasting of dynamical systems.

References

D.J. Acheson. *Elementary Fluid Dynamics*. Oxford University Press, 1989.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. NIPS'15, pp. 1171–1179, Cambridge, MA, USA, 2015. MIT Press.

Emmanuel de Bezenac, Arthur Pajot, and Patrick Gallinari. Deep learning for physical processes: Incorporating prior scientific knowledge. In *ICLR*, 2018.

Bo Chang, Lili Meng, Eldad Haber, Lars Ruthotto, David Begert, and Elliot Holtham. Reversible Architectures for Arbitrarily Deep Residual Neural Networks. In *AAAI*, pp. 2811–2818, 2018a.

Bo Chang, Lili Meng, Eldad Haber, Frederick Tung, and David Begert. Multi-Level residual Networks from Dynamical System View. In *ICLR*, pp. 1–14, 2018b.

Weinan E. A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, 5:1–11, 02 2017.

Aidan N. Gomez, Mengye Ren, Raquel Urtasun, and Roger B. Grosse. The Reversible Residual Network: Backpropagation Without Storing Activations. In *NIPS*, 2017.

Simon S. Haykin. *Neural networks and learning machines*. Pearson Education, third edition, 2009.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 770–778, 2016.

Byungsoo Kim, Vinicius C. Azevedo, Nils Thuerey, Theodore Kim, Markus Gross, and Barbara Solenthaler. Deep Fluids: A Generative Network for Parameterized Fluid Simulations. pp. 1–12, 2018.

Julia Ling, Andrew Kurzawski, and Jeremy Templeton. Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *Journal of Fluid Mechanics*, 807:155–166, November 2016.

J L Lions, R Temam, and Shouhong Wang. On the equations of the large-scale ocean. *Nonlinearity*, 5(5):1007, 1992.

Zichao Long, Yiping Lu, Xianzhong Ma, and Bin Dong. PDE-Net: Learning PDEs from Data. In *ICML*, pp. 3214–3222, 2018.

Yiping Lu, Aoxiao Zhong, Quanzheng Li, and Bin Dong. Beyond Finite Layer Neural Networks: Bridging Deep Architectures and Numerical Differential Equations. In *ICML*, pp. 3282–3291, 2018.

Mark M. Meerschaert and Charles Tadjeran. Finite difference approximations for two-sided space-fractional partial differential equations. *Applied numerical mathematics*, 56(1):80–90, 2006.

Barak A. Pearlmutter. Gradient calculations for dynamic recurrent neural networks: a survey. *IEEE Trans. Neural Networks*, 6(5):1212–1228, 1995.

Phillip James Edwin Peebles. *The large-scale structure of the universe*. Princeton university press, 1980.

Maziar Raissi. Deep hidden physics models: Deep learning of nonlinear partial differential equations. *Journal of Machine Learning Research*, 19(25):1–24, 2018.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. URL `http://arxiv.org/abs/1505.04597`.

Samuel H. Rudy, Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Data-driven discovery of partial differential equations. *Science Advances*, 3(4):e1602614, April 2017.

Lars Ruthotto and Eldad Haber. Deep Neural Networks motivated by Partial Differential Equations. 2018. URL `http://arxiv.org/abs/1804.04272`.

Andrew M. Saxe, James L. McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks, 2013. URL `http://arxiv.org/abs/1312.6120`. cite arxiv:1312.6120.

Russell Stewart and Stefano Ermon. Label-Free Supervision of Neural Networks with Physics and Domain Knowledge. *arXiv:1609.05566 [cs]*, sep 2016. URL `http://arxiv.org/abs/1609.05566`.

Nils Thuerey and Tobias Pfaff. MantaFlow. 2018. URL `http://mantaflow.com`.

Ah Chung Tsoi and A. D. Back. Locally recurrent globally feedforward networks: a critical review of architectures. *IEEE Transactions on Neural Networks*, 5(2):229–239, March 1994. ISSN 1045-9227. doi: 10.1109/72.279187.

8 APPENDIX

8.1 NAVIER-STOKES AND EULER EQUATIONS

Let us consider a fluid blob of surface δS, velocity u and density ρ. This blob has a pressure p which exerts a force $-\nabla p \delta S$ and is in a gravitational field g so that Newton's second law gives :

$$\rho \delta S \frac{Du}{Dt} = (-\nabla p + \rho g)\delta S$$

where $\frac{Du}{Dt}$ expresses the acceleration of the blob thus being the temporal derivative of the function $t \to u(x(t), y(t))$ which can also be expressed as :

$$\frac{Du}{Dt} = \frac{\partial u}{\partial t} + (u \cdot \nabla)u$$

Moreover, we will also make the incompressibility hypothesis which can be mathematically translated to :

$$\nabla \cdot u = 0$$

meaning that u is of null divergence.

Finally, by definition, the mass of the blob we are studying is conserved so that :

$$\frac{D\rho}{Dt} = \frac{\partial \rho}{\partial t} + (u \cdot \nabla)\rho = 0$$

Those equations are the so called *Euler equations* and form the system :

$$\begin{cases} \frac{\partial u}{\partial t} + (u \cdot \nabla)u = -\frac{\nabla p}{\rho} + g \\ \frac{\partial \rho}{\partial t} + (u \cdot \nabla)\rho = 0 \\ \nabla \cdot u = 0 \end{cases}$$

When deriving the Euler equations, we never considered the particular nature of the fluid. This is taken into account by considering an additional force in the previous derivation : viscosity. Even for small values, this additional force is of great importance : For example, while air usually has quite small viscosity, without it one wouldn't be able to explain why planes fly.

For newtonian fluids[3], viscosity can be taken into account with an additional term $\nu \nabla^2 u$ where $\nu = \frac{\mu}{\rho}$ is the *kinematic viscosity*. This gives us the following equation system :

$$\begin{cases} \frac{\partial u}{\partial t} + (u \cdot \nabla)u = -\frac{\nabla p}{\rho} + g + \nu \nabla^2 u \\ \frac{\partial \rho}{\partial t} + (u \cdot \nabla)\rho = 0 \\ \nabla \cdot u = 0 \end{cases}$$

Those are the *Navier-Stokes equations* which are central to fluid dynamics. We can easily see how the Euler equations can be considered as a simplification of this system, found by taking $\nu = 0$.

The importance of viscosity in a fluid, and thus a way to assess its qualitative behaviour, cannot be measured through ν alone as it has to be compared against the importance of advection. The most commonly used quantity to describe it is the *Reynolds number*. If we take U to be a typical speed and L a typical distance of the studied flow then it can be expressed as :

$$R = \frac{UL}{\nu}$$

[3]Some fluids can behave quite differently and this is translated into a different viscosity term

To understand its meaning, let us recall that we have $(u \cdot \nabla)u \sim \frac{U^2}{L}$ and $\nu \nabla^2 u \sim \frac{\nu U}{L^2}$ so that :

$$R \sim \frac{(u \cdot \nabla)u}{\nu \nabla^2 u}$$

which means that R is the ratio of the inertia and the viscosity terms. Thus, if R is small, it means that the fluid is very viscous, thus behaving more like honey ; If R is large, the viscosity term is small and the fluid behaves more like water (which doesn't always mean that every such fluid can be modelled with the Eulerian approximation as viscosity can play a role even when the corresponding term is small).

8.2 Leray projection : Euler and Navier stokes expressed in the form of equation 1

The two systems equation 8.1 and equation 8.1 are not of the form equation 1 as we still have the pressure variable p as well as the null divergence constraint. Let us show how we can transform those equations in order to get rid of it.

The Helmholz-Leray decomposition result states that for any vector field a, there exists b and c such that :

$$a = \nabla b + c$$

and

$$\nabla \cdot c = 0$$

where $\nabla \cdot$ is the divergence operator. Moreover, this pair is unique up to an additive constant for b. Thus, we can define a linear operator $\mathbb{P}$ by :

$$\mathbb{P}(a) = c$$

This operator is a continuous linear projector which is the identity for divergence-free vector fields and vanishes for those deriving from a potential.

Let us take a solution of NS and apply $\mathbb{P}$ on the first equation of equation 8.1, we have, as u is divergence free from the third equation and as g derives from a potential :

$$\frac{\partial u}{\partial t} = -\mathbb{P}[(u \cdot \nabla)u] + \nu \mathbb{P}(\nabla^2 u)$$

where permuting derivation and $\mathbb{P}$ is justified by the continuity of the operator[4].

Thus, if u is solution to equation 8.1, it is also a solution to :

$$\begin{cases} \frac{\partial u}{\partial t} = -\mathbb{P}[(u \cdot \nabla)u] + \nu \mathbb{P}(\nabla^2 u) \\ \frac{\partial \rho}{\partial t} = -(u \cdot \nabla)\rho \end{cases}$$

which is of the form of equation 1

Conversely, the solution of the above system is such that :

$$u_t = \int \frac{\partial u}{\partial t} = \int -\mathbb{P}[(u \cdot \nabla)u] + \nu \mathbb{P}(\nabla^2 u)$$

which gives, by exchanging $\mathbb{P}$ and the integral[5] :

$$u_t = \mathbb{P}[\int -(u \cdot \nabla)u + \nu \nabla^2 u]$$

so that u is automatically of null divergence by definition of $\mathbb{P}$. The two systems are thus equivalent.

[4] One can use a finite difference approximation to show it for example.

[5] To prove this, we just have to take a sum approximation to the integral and use again the linearity then the continuity of $\mathbb{P}$.

	t_0+1	t_0+5	t_0+10	t_0+15	t_0+20	t_0+25	t_0+30
Scheduled Sampling	**0.007**	0.044	0.100	0.176	**0.235**	**0.284**	**0.327**
No Teacher Forcing	0.008	**0.043**	**0.108**	**0.175**	0.236	0.289	0.335
Systematic Teacher Forcing	0.026	0.215	0.108	0.567	0.663	0.736	0.795

Table 6: Forecasting with Jointly Trained system : Test average MSE per time-step with Scheduled Sampling, Systematic Teacher Forcing and No Teacher Forcing, for different forecasting horizons.

	t_0+1	t_0+5	t_0+10	t_0+15	t_0+20	t_0+25	t_0+30
Scheduled Sampling	0.011	0.06	**0.102**	**0.179**	**0.234**	**0.288**	**0.341**
No Teacher Forcing	0.011	**0.058**	**0.102**	0.182	0.24	0.294	0.346
Systematic Teacher Forcing	**0.009**	0.076	0.225	0.374	0.5	0.605	0.694

Table 7: Forecasting with Pretrained estimator : Test average MSE per time-step with Scheduled Sampling, Systematic Teacher Forcing and No Teacher Forcing, for different forecasting horizons.

8.3 Comparison of sampling strategies

Tables 6 and 7 compare the 3 sampling strategies introduced in section 3, namely systematic teacher forcing, no teacher forcing and scheduled sampling. For the latter, we consider a decreasing sequence $(\epsilon_i)_i$, starting from 1 and converging to 0, so that, at time i, we pick Y_{t-1} with probability ϵ_i and $\widehat{Y}_{t-1}$ with probability $1-\epsilon_i$. This amounts to starting with one-step predictions then gradually increasing the time horizon as the model improves. The $(\epsilon_i)_i$ sequence has to decay at the same pace as the error of the model and thus has to be chosen carefully depending on the trained model. In this work, we have considered the exponential scheme :

$$\epsilon_i = \alpha^i$$

teacher forcing performs poorly, surprisingly not using any teacher forcing works remarkably well, as well as scheduled sampling while being much faster, which might be used to accelerate training while keeping a good performance. In all the experiments, we however used scheduled sampling.

8.4 Additional figures

We provide below some example of forecasting images obtained for Navier Stokes equations. The setting of the experiments is introduced in the main text in section 5.

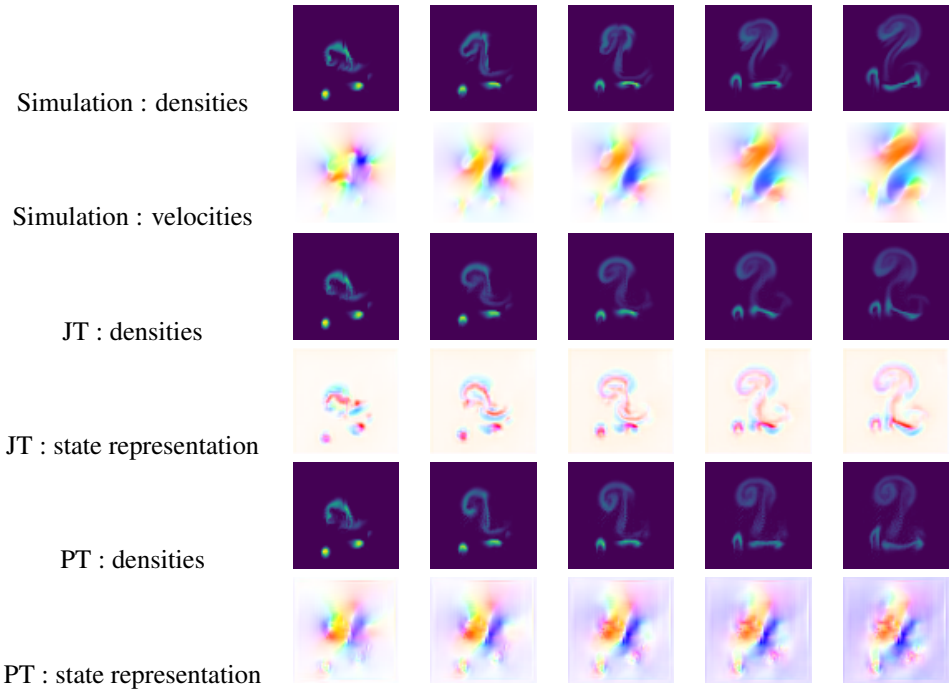

Figure 3: Predicted dynamics for the Navier Stokes equations with $R = 100000$ for $t + l = 4, 8, 12, 16, 20$, each algorithm having received the same 3 initial images (not shown here). Top 2 rows, correspond to the simulation which in our context is the groundtruth. First row is the generated density and second one is the velocity vector field. For the latter, each point is two dimensional, the color code is provided in the appendix. Rows 3 and 4 give the predictions obtained with a JT MSRE model, rows 5 and 6 the ones for the PT SST one.

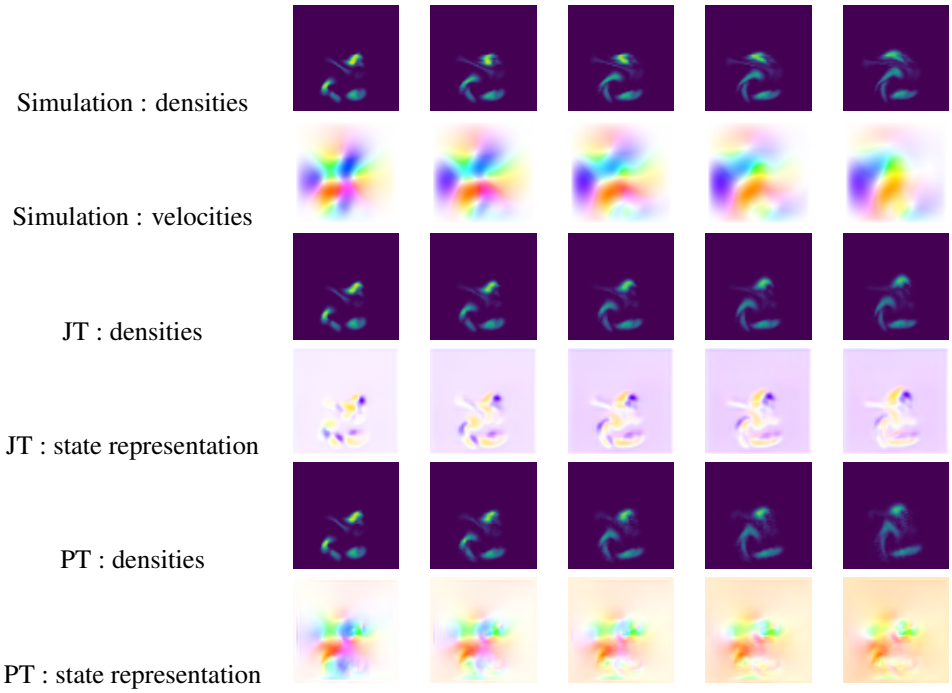

Figure 4: Predicted dynamics for the Navier Stokes equations with $R = 5000$ for $t + l = 4, 8, 12, 16, 20$, each algorithm having received the same 3 initial images (not shown here). Top 2 rows, correspond to the simulation which in our context is the groundtruth. First row is the generated density and second one is the velocity vector field. For the latter, each point is two dimensional, the color code is provided in the appendix. Rows 3 and 4 give the predictions obtained with a JT MSRE model, rows 5 and 6 the ones for the PT SST one..